AUT
LPG8 PG18 PG21 <>
REG <PG21 GRA

Eduardo Roreira

Definição formal dos autômatos - (V, S, q_0, F, d)

V = vocabulário S = conjunto de estados q_0 = estado inicial F = conjunto de estados finais
d = função de transição $d(q, a) = p$

Determinístico → cada estado tem só 1 transição p/ cada símbolo. Não-determinístico → quebra essa regra

ALG1 - Conversão de não-determinísticos em determinísticos:

*lembrar de ver o estado vazio nos rascunhos

1. Começa com o estado inicial ou concatenação de estados que se pode chegar sem usar algum símbolo (transições ε). Verifica então, para cada símbolo, quais estados são alcançáveis, e cria um estado concatenando todos eles. Ex: $d(q_0, a) = \{q_1, q_2\}$ → cria estado q_1q_2
2. Assim cada estado mantém suas transições determinísticas e faz as não-determinísticas p/ a concatenação encontrada
3. Para os estados compostos, as transições são as concatenações dos estados individuais que os formam

ALG2 - Minimizar estados de um autômato finito determinístico:

1. Separa os estados em dois grupos inicialmente: F (estados finais) e NF (estados não finais)
2. Para cada grupo, se analisa como cada estado se comporta p/ cada símbolo. Se houverem diferenças (erro, terminam em grupos diferentes, etc) precisam ser separados.
3. Repete até não poder mais separar os estados. O grupo com q_0 é o novo inicial e aqueles com finais são finais

gramática → variáveis terminais (símbolos do vocabulário) e não-terminais (geram produções)
→ tipos: sem restrições. Ex: aAbB → abab ; sensíveis ao contexto: produção do que produzidos ↑ menos variáveis na origem da mesma qtd EX: AB → ab | ac ;
livres de contexto: só um símbolo na origem da produção. EX: A → B x A
regulares: só um não-terminal na produção e um terminal produzido, podendo ter um não-terminal acompanhando EX: A → aA | aB | a

ALG3 - Remoção de recursão à esquerda:

1. Remove recursões indiretas substituindo símbolos não-terminais por suas produções em outras produções não-terminais
2. Remove recursões à esquerda ao separar a expressão em alfas (com recursão à esquerda) e betas (demais). Cria então duas produções: uma sendo a união dos betas seguidos da nova variável não-terminal e outra a união dos alfas com ε sendo os alfas seguidos de recursão da nova produção. Ex: S → Aa, A → Sb | Ac | d ⇝ A → Sb A' | dA', A' → cA' | ε

ALG4 - Remoção de ambiguidade (uma sentença pode ser derivada de mais de uma maneira pela gramática):

1. Cria uma produção p/ cada nível de prioridade, colocando as de maior prioridade antes (topo da árvore) e menor prioridade depois. As de menor prioridade serão então calculadas antes p/ poder calcular os demais. Ex:

~~associativos à esquerda: $E_i \to E_i$ op E_{j+1} (como o igual)~~
~~à direita: $E_i = E_j$ op E_i (como o diferente)~~

$E_4 \to E_3 = E_4 \mid E_4 <> E_3 \mid \dots \mid E_3$ ⇝

$E_3 \to E_3 + E_2 \mid E_3 - E_2 \mid \dots \mid E_2$

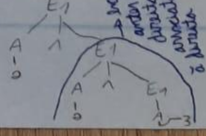
$E_2 \to E_2 * E_1 \mid E_2 / E_1 \mid \dots \mid E_1$

$E_1 \to$ not $E_1 \mid \underline{A \wedge E_1} \mid A$

$A \to 0 \mid 1 \mid 2 \mid \dots \mid 9$

se vem antes, fica embaixo

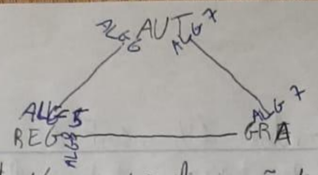
2^{2^3}



2+6+3 → associativo à esquerda

$E_1 \to$ not E_1 → unário

não associativos.
$E_1 \to E_0$ op E_0

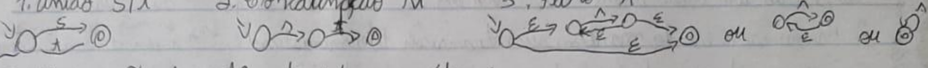


ALG 5 - Conversão de autômato p/ regex por eliminação de estados:

1. Isola os estados inicial e final, (i) se houver estados chegando ou (f) saindo.
2. Um por um, elimina os estados, criando novas transições com todas as combinações de entrada e saída que o teriam como intermediário. Fechos são incluídos nas novas transições no meio da concatenação da entrada e saída.

ALG 6 - Conversão de regex p/ autômato por Thompson

1. União s|t 2. Concatenação st 3. Fecho s*



ALG 7 - Conversão de autômato p/ gramática:

1. Cada estado do autômato é uma produção e símbolo não-terminal da gramática
2. Transições tem o símbolo passado como símbolo terminal e o estado que recebe é um não-terminal concatenado. Se for uma transição p/ um estado final, não-terminal pode ser desprezado. Gramática gerada é regular.

ALG 8 - Conversão de gramática regular para regex:

1. Cada produção vira um regex, e não terminais são sendo substituídos até o estado inicial ser um regex único.
2. Quando não há não-terminais (fora o próprio da produção) pode gerar o regex ao concatenar o fecho da união dos símbolos recursivos (removendo a recursão) com a união dos não recursivos

$E_3 \to E_2 < E_2 \mid E_2 > E_2 \mid E_2$

$E_2 \to E_2 + E_1 \mid E_2 - E_1 \mid E_1$

$E_1 \to E_1 * E_0 \mid E_1 / E_0 \mid E_0$

$E_0 \to C \,\hat{}\, E_0 \mid C$

$C \to cnst \mid id \mid (E_3) \mid id(E)$

$E \to E_3 L \mid \varepsilon$

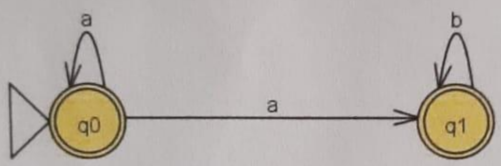
$L \to , E \mid \varepsilon$

Universidade de Caxias do Sul
FBX4064A - Linguagens Formais
Prof. Ricardo Dorneles
Avaliação da Primeira Área - 16/05/25

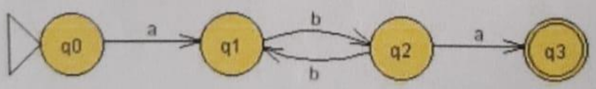
1)(1 ponto) Escreva uma expressão regular sobre o alfabeto $\{a, b, c\}$ que gere a linguagem cujas sentenças contenham exatamente duas ocorrências do símbolo **a**.

2)(1 ponto) Encontre um autômato finito determinístico que reconheça a linguagem gerada pela expressão $a^*(a|\epsilon)b^*$.

3)(1.5 pontos) Converta o autômato a seguir para um autômato finito determinístico equivalente.



4)(1.5 pontos) Encontre a expressão regular equivalente ao autômato a seguir:



5)(1 ponto) Converta a gramática regular a seguir para uma expressão regular. Considere que o símbolo inicial da gramática é o símbolo S:

S → 1S|2S|0C

B → 0B|1C|2S

C → 0B | ϵ

6)(1 ponto) Elimine a recursão à esquerda (direta e indireta) da gramática a seguir:

S → A0 | 0B | 11

A → 01 | S0

B → B0 | 0A

7)(1.5 pontos) Construa uma gramática livre de contexto que gere a linguagem $L = \{wbw^R\}$ onde w representa qualquer sentença sobre $\{a,b\}^*$ e w^R representa a sentença inversa a w, ou seja, os mesmos símbolos de w na ordem contrária. Assim, se w=abb, w^R=bba.

$\{a,b\}^* b \{a,b\}^{*R}$

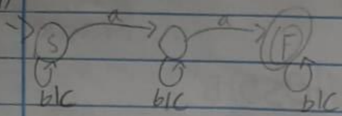
8)(1.5 pontos) Apresente uma gramática regular que gere a linguagem formada por todas as sentenças sobre $\{a,b,c\}^*$ onde o símbolo **a** não ocorre imediatamente à esquerda de um símbolo **b**.

Eduardo Pereira

dez

___/___/___

1)

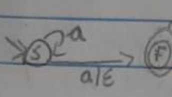


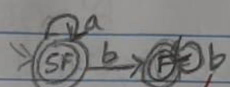
R: $(b|c)^* a (b|c)^* a (b|c)^*$ 1,0

a

abbcbbcabcb

2) $a^* (a|\varepsilon) b^*$





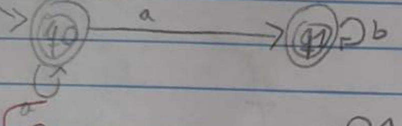
(SF, b) = {F}

(SF, a) = {SF}

(F, b) = {F}

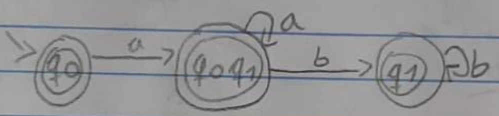
1,0

3)



$a^* b^*$

1,5



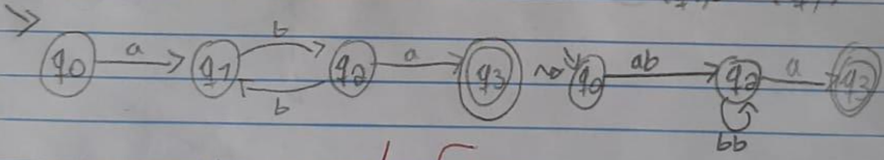
$(q_0, a) = \{q_0, q_1\}$

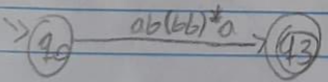
$(q_0q_1, b) = \{q_1\}$

$(q_0q_1, a) = \{q_0, q_1\}$

$(q_1, b) = \{q_1\}$

4)





1.5

5) $S \to 1S | 2S | 0C$ ⇝ $S \to 1S | 2S | 00B | 0$

$B \to 0B | 1C | 2S$ ⇝ $B \to 0B | 10B | 1 | 2S$ ⇝ $B \to (0|10)^* (1|2S)$

$C \to 0B | \varepsilon$

$S \to 1S | 2S | 00(0|10)^*(1|2S) | 0 \to 1S | 2S | 00(0|10)^*1 | 00(0|10)^*2S | 0$

$S \to (1|2|00(0|10)^*2)^* (00(0|10)^*1 | 0)$

1,0

000100 0100000 A0 S00 A000 00110000

0B00 → 00A00 000100 S0000 0100000 0B00 00A00 00S000

6) $S \to A0 \mid 0B \mid 11 \leadsto S \to 010 \mid S00 \mid 0B \mid 11$

$A \to 01 \mid S0$

$B \to B0 \mid 0A \leadsto B \to 0AB' \leadsto B \to (001 \mid 0S0)B'$

$B' \to 0B' \mid \varepsilon$

$S \to 010 \mid S00 \mid 0001B' \mid 00S0B' \mid 11$

R: $S \to 010S' \mid 0001B'S' \mid 00S0B'S' \mid 11S'$

$S' \to 00S' \mid \varepsilon$

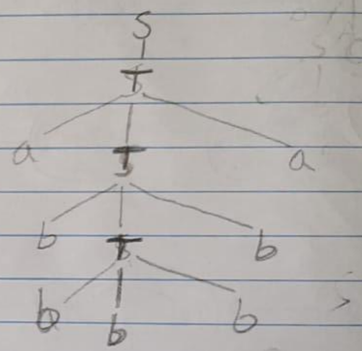
$B' \to 0B' \mid \varepsilon$

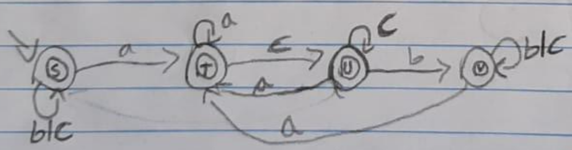
1.0

7) $S \to T \mid b$

$T \to aTa \mid bTb \mid aba \mid bbb$

1.5



8) 

$S \to aT \mid bS \mid cS \mid \varepsilon$

$T \to cU \mid aT \mid \varepsilon$

$U \to bV \mid cU \mid aT \mid \varepsilon$

$V \to bV \mid cV \mid aT \mid \varepsilon$

1.5